UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/740,661	12/18/2000	Hong Zhang	22271-05234	3497

758 7590 10/04/2003

FENWICK & WEST LLP
SILICON VALLEY CENTER
801 CALIFORNIA STREET
MOUNTAIN VIEW, CA 94041

EXAMINER	
PATEL, HARESH N	
ART UNIT	**PAPER NUMBER**
2126	4

DATE MAILED: 10/04/2003

Please find below and/or attached an Office communication concerning this application or proceeding.

DRG

Office Action Summary	Application No.	Applicant(s)	
	09/740,661	ZHANG ET AL.	
	Examiner	Art Unit	
	Haresh Patel	2126	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☐ Responsive to communication(s) filed on _____ .

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-35 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-35 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on 18 December 2000 is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

11)☐ The proposed drawing correction filed on _____ is: a)☐ approved b)☐ disapproved by the Examiner.

If approved, corrected drawings are required in reply to this Office action.

12)☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. §§ 119 and 120

13)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____ .

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).

a) ☐ The translation of the foreign language provisional application has been received.

15)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☒ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449) Paper No(s) 3 .
4) ☐ Interview Summary (PTO-413) Paper No(s). _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: .

DETAILED ACTION

1. Claims 1-35 are presented for examination.

Specification

2. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.

The following title is suggested: "A method to reduce stack memory resources in a threaded computer system that executes concurrent user sessions of the scalable network".

3. Applicant is reminded of the proper content of an abstract of the disclosure.

A patent abstract is a concise statement of the technical disclosure of the patent and should include that which is new in the art to which the invention pertains. If the patent is of a basic nature, the entire technical disclosure may be new in the art, and the abstract should be directed to the entire disclosure. If the patent is in the nature of an improvement in an old apparatus, process, product, or composition, the abstract should include the technical disclosure of the improvement. In certain patents, particularly those for compounds and compositions, wherein the process for making and/or the use thereof are not obvious, the abstract should set forth a process for making and/or use thereof. If the new technical disclosure involves modifications or alternatives, the abstract should mention by way of example the preferred modification or alternative.

The abstract should not refer to purported merits or speculative applications of the invention and should not compare the invention with the prior art.

Where applicable, the abstract should include the following:
(1) if a machine or apparatus, its organization and operation;
(2) if an article, its method of making;
(3) if a chemical compound, its identity and use;
(4) if a mixture, its ingredients;
(5) if a process, the steps.

Extensive mechanical and design details of apparatus should not be given.

The abstract of the disclosure is objected to because it does not contain computer terminology. Key terms involved in the invention like Internet, user session, network device, Java are missing in the abstract. Correction is required. See MPEP § 608.01(b).

Drawings

4. A new corrected drawing is required in this application because block 920 of Figure 9 contains "stock" instead of "stack". The corrected drawing is required in reply to the Office action to avoid abandonment of the application. The requirement for corrected drawings will not be held in abeyance.

Information Disclosure Statement

5. An initialed and dated copy of Applicant's IDS form 1449, Paper No. 3, is attached to the instant Office action.

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 1-35 are rejected under 35 U.S.C. 103(a) as being unpatentable over Nilsen et. al. 6,081,665 (Hereafter Nilsen) in view of Belkin et. al 6,542,920 (Hereafter Belkin).

8. As per claims 1, 9, 18 and 33-35, Nilsen teaches the following:

a computer implemented method of allocating stack memory for a process for executing a computer program code, the method comprising,

a computer system having an operating system for concurrently executing a plurality of user session requests, comprising,

a method of reducing stack memory resources in a computer system that executes concurrent user sessions, the method comprising,

a method of programming a computer program user code for execution by a thread in a threaded computer system, the method comprising,

a computer readable medium including program code for execution of a process in a computer system, the computer system having at least one computer thread having a first stack memory having a first stack size allocated to the thread and an alternate stack memory space having a second stack size, the program code comprising,

a computer thread for executing program code, comprising:

mapping an active session to a thread for execution (e.g., the mapping from method name and signature to index position is defined by the class loader, as described in "The Java Virtual Machine Specification", by Lindholm and Yellin, 1996, Addison-Wesley, col. 13, lines 24 - 64), the thread having a first stack memory selected to execute a first class of code (e.g., Stack overflow checking and expansion is greatly simplified by the presence of MMU hardware. Each of the three stacks associated with every thread can be represented as a number of virtual memory pages with only the first page initially allocated and all other pages unallocated and marked as inaccessible. When the corresponding stack expands into the unallocated region the

fault handler allocates and maps a new stack page. Once allocated, stack pages are not discarded until the thread terminates, col. 37 lines 1-8),

responsive to a code segment of the code being of the first class (e.g., the C stack holds C-declared local variables and run-time state information associated with compiler generated temporaries, col. 39 line 1 – col. 42 line 67), executing the code segment with the first stack memory (e.g., the C stack holds C-declared local variables and run-time state information associated with compiler generated temporaries, col. 12 line 56 – col. 13 line 8); and

responsive to the code segment being of a second class (e.g., the PERC pointer stack holds the pointer arguments passed as inputs to the method, pointer local variables, temporary pointers pushed during expression evaluation, and pointer values pushed as arguments to methods called by the current method, col. 12 line 21 – col. 14 line 45),

executing the code segment in an auxiliary stack memory to execute the code segment (e.g., The PERC non-pointer stack holds non-pointer arguments passed as inputs to the method, non-pointer local variables, temporary non-pointer values pushed during expression evaluation, and non-pointer values pushed as arguments to be called by this method, col. col. 12 line 21 – col. 14 line 45) and reclaiming the auxiliary stack memory subsequent to executing the code segment (e.g., Defragmenting Garbage Collection, as the term is used in this invention disclosure, describes a garbage collection technique that relocates in-use memory objects to contiguous locations so as to coalesce multiple segments of free memory into larger free segments, col. col. 12 line 21 – col. 14 line 45),

identifying function calls of the program code requiring stack memory greater than the stack memory allocated to the thread (e.g., Additionally, using the multiple-stack technique to

identify pointers on the run-time stack, the invention includes a method for performing efficient defragmenting real-time garbage collection using a mostly stationary technique, abstract), and

wrapping each function call requiring stack memory greater than that allocated to the thread with a wrapper configured to call an auxiliary stack memory to execute the function call (e.g., using the multiple-stack technique to identify pointers on the run-time stack, The invention also includes a method for efficiently mixing a combination of byte-code, native, and JIT-translated methods in the implementation of a particular task, where byte-code methods are represented in the instruction set of the virtual machine, native methods are written in a language like C and represented by native machine code, and JIT-translated methods result from automatic translation of byte-code methods into the native machine code of the host machine, abstract),

mapping an active session having a program code to a thread for execution, the thread having a first stack memory space allocated to the thread selected to handle a first class of function calls (e.g., The memory fault handler can either abort the thread because of stack overflow or it can enlarge the stack by mapping a newly allocated virtual memory page to the stack overflow address, col. col. 12 line 21 – col. 14 line 45),

transferring the execution of the program code from the first stack memory to an auxiliary stack memory having a stack memory size greater than the first stack memory responsive to the program code invoking a function call of a second class of function calls that requires a stack memory size greater than that of the first stack memory (e.g., The invention includes a method for implementing a single abstract virtual machine execution stack with multiple independent stacks in order to improve the efficiency of distinguishing memory pointers from non-pointers. Further, the invention includes a method for rewriting certain of the virtual

machine instructions into a new instruction set that more efficiently manipulates the multiple stacks, abstract),

reclaiming the auxiliary stack memory (e.g., Defragmenting Garbage Collection, as the term is used in this invention disclosure, describes a garbage collection technique that relocates in-use memory objects to contiguous locations so as to coalesce multiple segments of free memory into larger free segments, col. col. 12 line 21 – col. 14 line 45),

a computer program code having code segments of different code class (e.g., FIG. 53 provides the Java implementation of the TaskDispatcher class, col. 12 line 21 – col. 14 line 45), the code including a first code class that requires the first stack memory size and a second code class that requires the second stack memory size (e.g., The invention includes a method for implementing a single abstract virtual machine execution stack with multiple independent stacks in order to improve the efficiency of distinguishing memory pointers from non-pointers, abstract), and

a wrapper wrapping each code segment of the second class configured to transfer execution of the function to the alternate stack memory space (e.g., the invocation routine adjusts the stack and other state information as necessary in order to transfer control to the called method, col. 12 line 21 – col. 14 line 45),

a pool of threads, each thread having an associated stack memory having a first stack size (e.g., The invention includes a method for implementing a single abstract virtual machine execution stack with multiple independent stacks in order to improve the efficiency of distinguishing memory pointers from non-pointers, abstract),

a thread mapper mapping each user session onto one of the threads (e.g., the mapping from method name and signature to index position is defined by the class loader, as described in "The Java Virtual Machine Specification", by Lindholm and Yellin, 1996, Addison-Wesley, col. col. 12 line 21 – col. 14 line 45),

an auxiliary stack memory having a second stack size, the second stack size being larger than the first stack size (e.g., the size of the PERC pointer and non-pointer stacks is specified by compile-time macro definitions, col. 39 line 1 – col. 42 line 67),

a program code for executing one of the user sessions (e.g., the user thread that is scheduled for execution blocks. So the watchdog's sole responsibility is to notify the dispatcher that the application thread has gone to sleep. In response, the dispatcher will schedule another thread for execution, col. 39 line 1 – col. 42 line 67),

the code including at least one code segment characterized by a code class, the code classes including a first code class that requires the first stack memory size and a second code class that requires the second stack memory size (e.g., a method for efficiently mixing a combination of byte-code, native, and JIT-translated methods in the implementation of a particular task, where byte-code methods are represented in the instruction set of the virtual machine, native methods are written in a language like C and represented by native machine code, and JIT-translated methods result from automatic translation of byte-code methods into the native machine code of the host machine, abstract); and

a wrapper for each code segment of the second class configured to transfer execution of the function to the auxiliary stack memory (e.g., the invocation routine adjusts the stack and

other state information as necessary in order to transfer control to the called method, col. 39 line 1 – col. 42 line 67),

switchable auxiliary stack memory means for executing function calls of second class requiring a stack memory resource greater than the first stack memory (e.g., Additionally, using the multiple-stack technique to identify pointers on the run-time stack, the invention includes a method for performing efficient defragmenting real-time garbage collection using a mostly stationary technique, abstract) and reclaiming the stack memory resource when the function call of the second class is completed (e.g., Defragmenting Garbage Collection, as the term is used in this invention disclosure, describes a garbage collection technique that relocates in-use memory objects to contiguous locations so as to coalesce multiple segments of free memory into larger free segments, col. 39 line 1 – col. 42 line 67),).

However, Nilsen does not specifically show the usage of the additional auxiliary stack memory.

Belkin teaches the following:

Auxiliary stack (e.g., The additional storage column 220 specifies whether the threads in a particular thread pool have additional private storage associated therewith in addition to a stack. Some services (such as JAVA type services) require threads with additional private storage, col. 6, lines39 –51).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teachings of Nilsen with the teachings of Belkin in order to facilitate the additional stack for the thread to finish the task and to overcome the stack overflow problem.

9. As per claims 2-8, 10-17 and 19-32, Nilsen teaches the following:

the code segment includes a function call and code segments of the second class include a wrapper configured to call the auxiliary stack memory to execute the function call (e.g., figure 16 the C declaration of the structure used internal to the PERC implementation to represent Class objects. Each Class object represents the definition of a particular programmer-defined type, col. 7, lines 35-38),

the thread is non-preemptive, the auxiliary stack memory is a shared stack, and the wrapper performs the operations of (e.g., one of the implemented threads is a garbage collection thread that operates asynchronously thereby resulting in the garbage collection thread being interleaved with other threads in arbitrary order, objects subject to garbage collection being either finalizable or non-finalizable, a finalizable object being subject to an action that is performed when the memory space allocated to the finalizable object is reclaimed by the garbage collection thread, the finalizing action being specified by including a non-empty finalizer method in the class definition, the garbage collection thread being able to distinguish a thread's pointer variables from the thread's non-pointer variables, preemption of a thread being allowed only if the thread is in a state identified as a preemption point, a thread being allowed to hold pointers in variables between preemption points that may not be visible to the garbage collection thread, pointer variables that may not be visible to the garbage collection thread being called fast pointers, pointer variables that are visible to the garbage collection thread being called slow pointers, each LLL, function being identified as either preemptible or non-preemptible, col. 67, lines 1 –21),

saving a stack pointer to the first stack (e.g., Within the called method, the frame pointer (fp) is adjusted to point at the memory immediately above the first pushed argument and the stack pointer (sp) is adjusted to make room for local variables to be stored on the stack, col. 6, lines 49 –67),

resetting the stack pointer to the shared stack (e.g., Within the called method, the frame pointer (fp) is adjusted to point at the memory immediately above the first pushed argument and the stack pointer (sp) is adjusted to make room for local variables to be stored on the stack, col. 6, lines 49 –67),

copying arguments from the first stack to the shared stack (e.g., including an "indirect pointer" field for each object in memory, the "indirect pointer" field containing a pointer to the location of the currently valid copy of the data that corresponds to the object, the pointer pointing to the object itself for objects in a mark-and-sweep space, the pointer pointing to the location of the object that currently represents the object's contents for objects in to-space and from-space, col. 73, lines 5-28),

calling a program function of the function call (e.g., To deallocate this memory, call freeCS(), passing as its single argument the void * that was returned by allocCS(), col. 6, lines 20-45),

returning the result to the first stack of the thread (e.g., leaving the return address in the stack slot above the top-of-stack entry on the non-pointer stack, figure 94); and

returning the shared stack (e.g., leaving the return address in the stack slot above the top-of-stack entry on the non-pointer stack, figure 94),

the thread is preemptive, the auxiliary stack is a new stack from a pool of stacks, and the wrapper performs the operations of

saving a stack pointer to the first stack memory (e.g., Within the called method, the frame pointer (fp) is adjusted to point at the memory immediately above the first pushed argument and the stack pointer (sp) is adjusted to make room for local variables to be stored on the stack, col. 6, lines 49 –67),

saving the stack address of the new stack segment (e.g., Within the called method, the frame pointer (fp) is adjusted to point at the memory immediately above the first pushed argument and the stack pointer (sp) is adjusted to make room for local variables to be stored on the stack, col. 6, lines 49 –67),

resetting the stack pointer to the new stack segment (e.g., Within the called method, the frame pointer (fp) is adjusted to point at the memory immediately above the first pushed argument and the stack pointer (sp) is adjusted to make room for local variables to be stored on the stack, col. 6, lines 49 –67),

returning the result of the program function to the first stack memory (e.g., To deallocate this memory, call freeCS(), passing as its single argument the void * that was returned by allocCS(), col. 6, lines 20-45),; and

returning the new stack segment (e.g., Within the called method, the frame pointer (fp) is adjusted to point at the memory immediately above the first pushed argument and the stack pointer (sp) is adjusted to make room for local variables to be stored on the stack, col. 6, lines 49 –67),

each of the classes includes a code type that is blockable and a code type that is non-blockable (e.g., Finally, the invention includes a method to analyze and preconfigure virtual memory programs so that they can be stored in ROM memory prior to program, abstract),

the code types are identified by a naming convention (e.g., class name)

However, Nilsen does not specifically show the usage of the additional auxiliary stack memory.

Belkin teaches the following:

Auxiliary stack (e.g., The additional storage column 220 specifies whether the threads in a particular thread pool have additional private storage associated therewith in addition to a stack. Some services (such as JAVA type services) require threads with additional private storage, col. 6, lines 39 –51),

allocating a preselected stack memory space for the auxiliary stack memory (e.g., The additional storage column 220 specifies whether the threads in a particular thread pool have additional private storage associated therewith in addition to a stack. Some services (such as JAVA type services) require threads with additional private storage, col. 6, lines 39 –51),

allocating the stack memory for the auxiliary stack memory space as required to satisfy the stack memory requirements of the function call (e.g., The additional storage column 220 specifies whether the threads in a particular thread pool have additional private storage associated therewith in addition to a stack. Some services (such as JAVA type services) require threads with additional private storage, col. 6, lines 39 –51),

allocating a new stack segment having a stack address (e.g., The additional storage column 220 specifies whether the threads in a particular thread pool have additional private

storage associated therewith in addition to a stack. Some services (such as JAVA type services) require threads with additional private storage, col. 6, lines 39 –51).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teachings of Nilsen with the teachings of Belkin in order to facilitate the additional stack for the thread to finish the task and to overcome the stack overflow problem.

Conclusion

10. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

See attached Form PTO-892.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Haresh Patel whose telephone number is (703) 605-5234. The examiner can normally be reached on Monday-Friday from 8:00 am to 5:30 pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, John Follansbee, can be reached at (703) 305-8498.

The appropriate fax phone number for the organization where this application or proceeding is assigned is (703) 306-5404.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

Haresh Patel

September 22, 2003.

JOHN FOLLANSBEE
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100